D4A Magazine

مجلة منتدى دلفي للعرب

العدد الخاص – خارج السلسلة لشهر رمضان



www.delphi4arab.com

فهـــرس العدد



افتتاحيـــــــــــــــــــــــــة



بمناسبة حلول شهر التوبة و الغفران تتقدم إدارة منتدى دلفي للعرب بتهانيها إلى كل أعضاء المنتدى و تتمنى لهم صوما مقبولا و ذنبا مغفورا.

و تعلن عن:

- بداية نشر دروس يومية من تقديم الأعضاء المشرفين على أقسام المنتدى، كل يوم من أيام رمضان سوف تطلعون إن شاء الله على درس جديد يعالج موضوع من مواضيع البرمجة بدلفي.
- فتح باب الترقية للأعضاء النشطين و المتميزين لتمكينهم من دخول قسم المتميزين.
- الباب مفتوح لكل عضو يرى في نفسه القدرة و العزم لتقديم دروس مفيدة و سهلة الهضم يتم تثبيتها و ترقية صاحبها إلى رتبة ـ مبرمج محترف ـ دون حساب عدد مشاركاته.

الكاتب: إدارة المنتدى

بقلم خالد الشقروني



برغم ظهور مفاهيم المنحى للكائن Object Orientation منذ الستينيات إلا أن اعتماد مفاهيم المنحى للكائن في تطبيقات الأعمال وقواعد البيانات لم يبدأ ظهورها إلا منذ أواخر الثمانينات. البرمجة بالمنحى للكائن تعتمد على تصميم الكود وفق عناصر أو كائنات قائمة بذاتها وشبه معزولة هذه العناصر تحمل في داخلها بياناتها وإجراءاتها، معا، مما يجعل من إنشاء وإدارة الكود البرمجي أكثر مرونة، وصيانته وإصلاح أخطائه أكثر سهولة، كما يصير أكثر قابلية للتوسع.

أيضا أتاح مفهوم المنحى للكائن ولأول مرة فرصة النظر إلى العناصر المكونة لعمليات التطوير البرمجي بصورة موحدة وشبه متطابقة. إذا سلمنا بأن تطبيقات الأعمال تشمل المستويات/المراحل التالية: الواقع الملموس، ثم وصفه ، ثم إعداد تصاميم لبرمجتها لاحقا ، ثم تنفيذ وبرمجة التصاميم، و تخزين البيانات.

فإن المنحى للكائن أتاح فرصة النظر إلى النقاط السابقة بمنظار موحد. كالتالي:

الواقع المعاش (أفراد، جهات، أماكن) = أشياء = كائنات

تحليل ورسم هذا الواقع = وصف الكائنات في الواقع

تصميم برمجي محاكي للواقع = ترجمة كائنات الواقع من خلال التحليل السابق إلى كائنات في بنيان برمجي.
تخزين بيانات الكائنات في قواعد بيانات تحاكي التصميم البرمجي.
وبصورة أقل : تصميم واجهات الاستخدام في البرامج انطلاقا من زاوية رؤية الواقع.

فتكون الدورة الحياتية بأبسط مكوناتها لتطبيق برمجي كالتالي:

الواقع المعاش:

أن يكون هناك واقعا ملموسا ومعاشا يراد التصدي له برمجيا لأي غرض كان. كأن تكون هناك مدرسة وبها معلمون وطلبة وفصول دراسية و مواد تدريس وجملة من العلاقات تربط بين كل واحدة منها بالآخر.

تحليل الواقع ورسم حدوده وتفكيكه إلى كائنات وعلاقات:

معلم	طالب	مادة دراسية:
-----	------	-----------
الاسم	رقم القيد: أرقام وحروف	الرقم: رقم
تاريخ الميلاد	الاسم	اسم المادة:
	تاريخ الميلاد: يوم/شهر/سنة	عدد الساعات:

ترجمة التحليل السابق إلى تصميم برمجي

TCourse: TClass	TStudent: TClass	TProfessor: TClass
-----------------	-----------------	-----------------
CNo: integer	RegNo: string	Name: string
Name: string	Name: string	BDate: TDate
Hours: integer	BDate: TDate	PhonNo: string
....	....	
AddStudent(TStudent)	GetCourses:TCourseList	

تخزين بيانات الكائنات واسترجاعها

عند تخزين بيانات الكائنات واسترجاعها وباستخدام قواعد البيانات الكائنية يتم الأمر بصورة مباشرة وبدون تفصيلات، مثلا عند تخزين كائن طالب، تتم العملية بتعليمة مثل:

db.Set(Student) **فتقوم قاعدة البيانات تلقائيا بحفظ كائن الطالب وحفظ جميع الكائنات المرتبطة به كالمواد الدراسية وغيرها.**

كما يتم استرجاع الكائن بنفس السهولة و المباشرة بتعليمة مثل:

db.Get(Student) ، **فيتم معها استرجاع جميع الكائنات المرتبطة به.**

في نظم قواعد البيانات الكائنية OODBMS **تكون قاعدة البيانات على علم بتصميم الكائنات وعلاقاتها، فلا حاجة لبناء الجداول و تحديد علاقات الربط بينها ، كما لا حاجة لتبيان أي جدول**

يحمل بيانات أي كائن. حيث تقوم بتخزين بيانات الكائنات وعلاقاتها مع الكائنات الأخرى بصورة مباشرة دون وسيط أو جسر برمجي. كما رأينا سابقا.

إلا أنه في الواقع العملي، معظم تطبيقات الأعمال الكائنية تعتمد أنظمة قواعد البيانات العلائقية، بدلا من قواعد البيانات الكائنية ، و ذلك لأسباب سنذكرها بعد حين.

مقاربة التصميم السابق إلى جداول في قاعدة بيانات علائقية

Course	Student	**Professor**
------	---------	--------
CId: integer	RegId: integer	TNo:integer<<PK>>
CNo: integer<<PK>>	RegNo: char(10)<<PK>>	PName: char(30)
CName: char(20)	SName: char(30)	BDate: date
Hours: integer	BDate: date	PhonNo: char(10)

لاحظ، الربط المباشر للكائنات بين التحليل والتصميم/الكود ومقاربة جداول البيانات.

(ملاحظة: التصاميم السابقة لتوضيح الفكرة فقط، ويعوزها الدقة والمتانة)

مفاهيم المنحى للكائن و قواعد البيانات

قواعد البيانات الكائنية لم تصل إلى المستوى الذي تحقق فيه معظم احتياجات الصناعة من متطلبات مناولة البيانات، كما تعوزها المواصفات القياسية و لغة استعلام موحدة، لذا كان استخدامها العملي محصورا في أنواع تطبيقات محدودة، التي تكون متطلباتها ضمن إمكانيات نظام قواعد البيانات الكائنية، أو تلك التي تكون بيانات الكائنات فيها متشعبة ومتشابكة كأن تحوي تكوين شجري متعدد المستويات من كائنات مختلفة بحيث يصعب تخزين بياناتها بنظم قواعد بيانات غير كائنية.

لذلك نجد أن تطبيقات الأعمال التي تعتمد المنحى الكائني تتجه نحو الاستعانة بنظم قواعد بيانات الأكثر رسوخا وثباتا، وهنا نقصد بالطبع أنظمة قواعد البيانات العلائقية RDBMS ، التي أثبتت مع مرور الزمن قوة ومتانة المفهوم الذي بني عليه تصميمها .

قواعد البيانات العلائقية

في مجال قواعد البيانات فإن المنتشر و السائد هي نظم إدارة قواعد البيانات العلائقية RDBMS التي أثبتت فعاليتها و إعتماديتها في حفظ البيانات و المحافظة على مصداقيتها، لكن بنيانها لا يدعم مفاهيم المنحى للكائن. وفي سبيل الحفاظ على متانة تصميم الكائنات في التطبيق و فصله عن العمليات الخاصة بالنفاذ لقاعدة البيانات وإبقاؤها معزولة في حيز خاص بها؛ يضطر القائمون على التطبيق البرمجي إلى إنشاء وسيط بين الكائنات و قواعد البيانات يتم فيه إنشاء المقاربات بين صنفيات الكائنات و جداول قواعد البيانات، كما تتم في هذا الوسيط أيضا عمليات ترجمة البيانات ونقلها بين قواعد البيانات والكائنات، والحرص قدر الإمكان على عدم إرهاق الكائنات وتحميلها أعباء إضافية تخص ديمومة البيانات وحفظها، وجعلها معزولة قدر الامكان عنها .

إقامة هذه الوسيط تتطلب جهدا إضافيا، خاصة إذا كان أعداد صنفيات الكائنات كبيرا، وعلاقاتها كثيرة ومتعددة المستوى و ما يتطلبه من إنشاء صنفيات وسيطة تحاكي صنفيات النظام ، وتقسيم جيد وتوزيع سليم لبيانات الكائنات في جداول قاعدة البيانات لضمان سلامة وصحة استرجاها مرة أخرى ككائنات بنفس الوضع الأصلي.

لذلك ظهرت أدوات وإطارات عمل تتصدى لهذه المهمة ORM وتجعل من أمور المقاربة بين الكائنات وقواعد البيانات العلائقية أكثر سهولة.

أدوات المقاربة ORM

أدوات المقاربة Object-relational mapping تختلف في مدى سهولة أو صعوبة تطويعها للربط بين الكائنات وقواعد البيانات، كما أنها تتفاوت فيما تقدمه من مزايا وخدمات، بعضها يوفر إمكانية إنشاء مخطط جداول قاعدة البيانات انطلاقا من الصنفيات classes الموجودة، وبعضها الآخر بالعكس من ذلك يرتكز على وجود مخطط لجداول قاعدة البيانات يعتمد عليها لإنشاء الصنفيات وتوليد التوليف أو الكود البرمجي، و بعضها يقدم الاثنين معا.

إلا أن لأدوات ORM سلبياتها أيضا. فهي وإن كانت توفر الوقت والجهد بالنسبة للتطبيقات الضخمة فإنها تشكل عبئا غير مبرر للتطبيقات الصغيرة و حتى المتوسطة، بما تفرضه من توصيفات، وتولده من صنفيات وسيطة وأكواد متشعبة، كما أنها تشكل قيدا غير مرن للمطورين الذين يضطرون للتعامل مع قواعد البيانات مباشرة بسبب عدم قدرة أو كفاءة الأدوات عند التعامل مع بعض الاستعلامات المعقدة.

أيضا أدوات ORM قد تفرض حلولا قد تؤدي إلى تصميم سيء لقاعدة البيانات، أو تنظيم غير محكم لكائنات النظام، زيادة على المراعاة المستمرة للمحافظة على التوافق الجيد والكفوء بين كائنات النظام وكائنات قاعدة البيانات أي الجداول كلما تتطور النظام أو تغيرت متطلباته.

لذلك فإن قرار استخدام أدوات ORM جاهزة أو عدم استخدامها يعتمد على النظرة المتوازنة لما ستقدمه من مزايا وما تفرضه من تضحيات في سياق النظام المستهدف.

ما هي أدوات ORM المتوفرة لبيئة دلفي

هذه بعض برمجيات ORM التي تستهدف التطبيقات المطورة بلغة دلفي أو ما يوافقها.

- tiOPF ، مصدر مفتوح
- hcOPF ، مصدر مفتوح
- g-framework ، مصدر مفتوح لتطبيقات الويب. آخر إصدار يدعم دلفي 2007.
- InstantObjects ، مصدر مفتوح، آخر إصدار يدعم دلفي 2006
- Synopse SQLite3 Framework ، مصدر مفتوح
- Delphi Spring Framework ، مصدر مفتوح
- Delphi Hibernate من فريق CnPack
- DObject
- ObjectSight
- Bold for Delphi ظهرت في 2002 وقامت بورلاند لاحقا في 2006 بشراء الشركة، و لم يظهر أي إصدار جديد لها منذ حينها، آخر نسخة تدعم دلفي 2007.

بقلم Kachwahed



يقوم نظام التشغيل Windows عادة برسم المكونات على النموذج عند تشغيل البرنامج بشكل تلقائي، ويسمح Windows بتخصيص الرسم على بعض المكونات لإعطائها واجهة أخرى...
عند ضبط خاصية OwnerDraw على القيمة True للمكون لن يقوم Windows برسم عناصر المكون. تسمح هذه الخاصية بتعديل شكل (أو مظهر) المكون من خلال التقاط حدث رسالة Windows التي يرسلها لكل عنصر من المكون.

يتلقى النموذج الرئيسي رسالة الحدث WM_DRAWITEM عند الحاجة إلى رسم شكل المكون، وحينها يقوم النموذج بتنفيذ إجراء مخصص لتخصيص شكل مختلف للمكون.
حسنا... لست بحاجة إلى معرفة كل هذه التفاصيل حيث تقوم دلفي بكل ذلك آليا، لنتحدث بلغة الأمثلة.

مثال بسيط لتوضيح الفكرة:

لنلاحظ تعريف الإجراء OnCustomDrawItem **للمكون** TListView **المخصص لتخصيص مظهر مختلف لكل سطر:**

```
procedure TForm1.ListViewFilesCustomDrawItem(Sender: TCustomListView;
  Item: TListItem; State: TCustomDrawState; var DefaultDraw: Boolean);
```

ما يهمنا هنا Item: TListItem **وهو يمثل أسطر المكون، ويملك خاصية** Index **التي تسمح بتحديد سطر معين، فمثلا لو كتبنا هذا:**

```
if Item.Index = 1 then Sender.Canvas.Brush.Color := clBlue;
```

لقمنا بتغيير لون ثاني سطر من الأسطر:

File Name	Size	Date
Text.txt	0	01/01/2010
Bitmap.bmp	20	02/01/2010
Jpeg.jpg	1	01/03/2010

وهنا مثلا نغير الأسطر بالتناوب بين الأزرق والأحمر:

```
with Sender.Canvas.Brush do //Sender يمثل المكون
 begin
   case Item.Index mod 2 of    //mod تعني تحسب باقي القسمة
     0: Color := clBlue;
     1: Color := clRed;
   end;
 end;
```

يملك النمط TListView **أحداثا أخرى مشابهة** OnCustomDraw **لتحديد مظهر مختلف للمكون ككل، و**OnCustomDrawSubItem **لتغيير مظهر التفاصيل (أو الأعمدة باعتبار طريقة العرض** vsReport **من الخاصية** ViewStyle **للمكون).**

لنأخذ مثال آخر أكثر فائدة:

نريد رسم الخطوط (Fonts) في مكون TListBox وليكن ListBoxFonts لتعبئته بأسماء الخطوط المتوفرة؛ يمكن قراءة محتوى المتغير Screen المعرف في المكتبة Forms.pas، مثال عند إقلاع النموذج (OnCreate) نكتب:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBoxFonts.Items := Screen.Fonts;
end;
```

الآن في الحدث OnDrawItem نرسم باستخدام Canvas المكون شكل الخط لكل سطر في المكون ListBoxFonts:

```
procedure TForm1.ListBoxFontsDrawItem(Control: TWinControl; Index: Integer;
  Rect: TRect; State: TOwnerDrawState);
begin
  with ListBoxFonts do
  begin
    Canvas.FillRect(Rect);                          {01}
    Canvas.Font.Name   := ListBoxFonts.Items[Index];   {02}
    Canvas.TextOut(Rect.Left, Rect.Top, ListBoxFonts.Items[Index]);
  end;
end;
```

الشرح:

{01} : عند رسم الأسطر (OnDrawItem) نقوم بتحديد إطار السطر Rect: TRect المعرف في الإجراء للرسم Canvas.

هنا نقوم بتحديد خط الكتابة للرسم Canvas.

وفي أخيرا نقوم برسم نص كل سطر وفق الخصائص التي حددناها في Canvas باستخدام TextOut.

الآن لقراءة اختيار الخط (Font) من المكون ListBoxFonts نكتب شيئا مثل:

```
TextLabel.Font.Name := ListBoxFonts.Items[ListBoxFonts.ItemIndex];
```

يمكن فعل ذلك مع أي مكون مشابه (TComboBox **مثلا)، أمثلة أخرى في المرفق.**

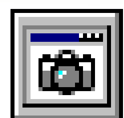
أفكار أخرى:

- **يمكن رسم مكونات أخرى على المكونات الأصلية من خلال تحديد الإطار (**TRect**) ثم الرسم على الأسطر بنفس الطريقة**
- **رسم صورة في خلفية المكون من خلال** Canvas **باستخدام** TBitmap
- **تخصيص شكل مختلف أو** Theme **باستخدام بعض توابع** API.

غير ذلك المجال مفتوح لمخيلتك J

بقلم Lam.Abdeldjalil





هو مكون من مكونات حزمة RX Library ، يسمح بقراءة و كتابة خصائص المكونات و حفظها في ملف من نوع ini أو في احد مفاتيح سجل النظام Windows Registry بدون كتابة أوامر إضافية من المستعمل ، يدعم جميع المكونات (الأصلية المثبتة مع دلفي افتراضيا أو المضافة)، باستعمال المكون لن تبقى عالقا في عملية تسيير عشرات ملفات ini.

طريقة عمل المكون:

تبدأ عملية المعالجة بعد الانتهاء من تحديد المكونات المراد حفظ خصائصها و تفعيل خيار الحفظ للمكون.

- في حدث إغلاق الفورم يقوم المكون بحفظ خصائص المكونات في ملف من نوع ini أو حفظها مباشرة في سجل النظام.
- في حدث إظهار الفورم يقوم المكون بقراءة قيم الخصائص المحفوظة سابقا.

خصائص TFormStorage :

خاصية UseRegistry : لحفظ الخصائص في سجل النظام، افتراضيا الخاصية ليست مفعلة أي أن المعلومات المتحصل عليها تحفظ في ملف ini.

خاصية Active : لتفعيل و إلغاء تفعيل الحفظ .

خاصية Active Control: لحفظ آخر مكون كان نشط قبل الإغلاق .

خاصية Form Position : لحفظ طول و عرض الفورم و مكان تواجدها في الشاشة.

خاصية Window State : لحفظ حالة الفورم wsNormal او wsMaximized .

مثال عن استعمالها :

ـ حفظ خصائص الفورم ، حالة الفورم و النص الموجود في مكون Edit1

1ـ ضع المكون TFormStorage على الفورم .

2ـ اختر اسم الملف ini الذي سوف تحفظ فيه الخصائص .

3ـ اضغط مرتين على المكون أو على خاصية StordProps ثم اضغط على (ثلاثة نقاط) … لاستدعاء FormStorage Designer.

4ـ اختر المكون المراد حفظ خاصية من خصائصه.

5ـ أختر الخاصية المراد حفظها.

6ـ اضغط على Add لإضافتها ثم اضغط على OK

انظر الشكل رقم 01.

ملاحظة: يمكن إضافة أكثر من خاصية لمكون واحد كما يمكن حفظ خصائص عدة مكونات في ملف ini واحد فقط.

لتحميل المكتبة اتبع الرابط: http://www.delphi4arab.com/forum/showthread.php?t=3479

شكل 01

بقلم STRELiTZIA



وحدة Printers.pas:

مسارها: X:\Program Files\Borland\Delphi7\Source\Vcl

استدعائها: uses printers;

مثال عملي للحصول على قائمة الطابعات المثبتة على النظام:

```
var PrintersList: TPrinter;
begin
  PrintersList := TPrinter.Create;
  Memo1.Lines := PrintersList.Printers;
  PrintersList.Free;
end;
```

وحدة StrUtils.pas:

مسارها: X:\Program Files\Borland\Delphi7\Source\Rtl\Common

استدعائها: uses StrUtils;

مثال عملي لعكس أحرف اسم معين:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Add(ReverseString('Delphi4Arab'));
end;
```

لنحصل على هذه النتيجة: barA4ihpleD

وحدة StrUtils تحتوي على مجموعة كبيرة من الدوال و الإجراءات متخصصة في معالجة المتغيرات من نوع String و عائلتها.

وحدة Math.pas:

مسارها: X:\Program Files\Borland\Delphi7\Source\Rtl\Common

استدعائها: uses Math;

مثال عملي لبعض العمليات الحسابية:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Add(FloatToStr(Power(2, 3)));
  Memo1.Lines.Add(FloatToStr(Min(2, 3)));
  Memo1.Lines.Add(FloatToStr(Max(2, 3)));
end;
```

وحدة Math.pas تحتوي على مجموعة كبيرة من الدوال و الإجراءات متخصصة في معالجة العمليات الحسابية.

وحدة DateUtils.pas:

مسارها: X:\Program Files\Borland\Delphi7\Source\Rtl\Common

استدعائها: uses DateUtils;

مثال عملي للحصول على تاريخ الأمس، اليوم و الغد:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.Lines.Add(DateToStr(DateUtils.Yesterday));
  Memo1.Lines.Add(DateToStr(DateUtils.Today));
  Memo1.Lines.Add(DateToStr(DateUtils.Tomorrow));
end;
```

وحدة DateUtils.pas تحتوي على مجموعة كبيرة من الدوال و الإجراءات متخصصة في معالجة العمليات الخاصة بالتواريخ.

الخلاصة:

دلفي مدجج بمكتبات جاهزة الاستعمال تعالج ميادين مختلفة من البرمجة، و اطلاعك عليها حتى و إن كان مجرد تصفح لإشباع حب الاستطلاع سوف يرجع عليك بفائدة كبيرة و يكون عنصر فعال في توسيع مفاهيمك البرمجية.